Towards Efficient Publish-Subscribe Middleware in the IoT with IPv6 Multicast

Sven Akkermans, Rafael Bachiller,
Nelson Matthys, Wouter Joosen, and Danny Hughes
iMinds-DistriNet, Department of Computer Science
KU Leuven, 3001 Leuven, Belgium
{name.surname}@cs.kuleuven.be

Mališa Vučinić
Grenoble Alps University
CNRS Grenoble Informatics Laboratory UMR 5217, France
STMicroelectronics, Crolles, France
Malisa.Vucinic@imag.fr

***Abstract*—Due to its scale and dynamism, the Internet of Things (IoT) requires efficient and flexible communication support. At the network layer, IPv6 integrates heterogeneous technologies to provide interoperability, efficient multicast group communication and a flexible address space. At the application layer, publish-subscribe (pub-sub) middleware implements a scalable, dynamic and loosely-coupled data dissemination scheme. The pub-sub paradigm is a natural use case for IPv6 multicast but the two mechanisms are poorly integrated in the IoT. We tackle this problem by proposing a framework that integrates pub-sub middleware and multicast to reduce communication overhead. Our solution maps application-layer subscriber groups to network-layer multicast groups. Pub-sub hosts can either implicitly derive the necessary multicast address or request it from a group manager. We evaluate our framework on an IoT network testbed composed of representative hardware and demonstrate improvements in bandwidth and energy consumption that scale with the size of the network. Bandwidth consumption of a publishing sensor decreases by up to 54% for 10 subscribers and 66% for 20 subscribers. Moreover, the implementation has a minimal memory footprint, requiring only an additional 1.3% dynamic memory and 4.7% flash storage.**

***Index Terms*—Internet of Things, multicast, IPv6, publish-subscribe**

I. Introduction

Recent years have seen an explosion of smart devices in all aspects of our society. These devices are no longer passive, but are increasingly interacting with the physical world. The resulting network with billions of interconnected objects is referred to as the Internet of Things (IoT). The IoT is partitioned into many IP-enabled networks that are composed of resource-constrained devices that can sense the physical environment and actuate appliances.

Because of the IoT scale, it is necessary to manage these devices in a scalable and flexible manner, *both* when it comes to interactions within the network and with the rest of the Internet. Furthermore, radio communications consume the majority of battery power and must therefore be minimised. While a range of point solutions have been developed to improve communication efficiency for the IoT, these are not generic or well integrated [1], [2].

On one hand, distributed *publish-subscribe* (pub-sub) middleware [3] implements a scalable, dynamic and loosely-coupled interaction scheme to disseminate data among IoT nodes. Subscribers express interest in information and are subsequently notified by an intermediate broker when a publisher, e.g., a sensor, generates such information. The intermediate broker allows nodes to be ignorant of network topology and each other, promoting manageability, decoupling, and scalability. On the other hand, IPv6 provides native support for efficient communication through *multicast*, a networking paradigm where a single packet transmission is delivered to multiple recipients. Hence, both pub-sub middleware and IPv6 multicast provide important features for the IoT, however, there is no systematic way to integrate these approaches [4].

This paper tackles this problem by building upon recent networking advancements, such as the IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) [5] and stateless multicast RPL forwarding (SMRF) [6], which allows efficient IPv6 multicast routing in a low-power network. We use LooCI [3] as a representative example of pub-sub middleware for the IoT. We address the gap between the application-layer pub-sub paradigm and the network-layer IPv6 multicast mechanism by proposing a framework that enables their systematic integration. In this context, our research addresses the following questions: (a) how can we integrate IPv6 multicast in existing IoT middleware; and (b) what are the benefits and downsides of integrating these mechanisms in the IoT.

The rest of the paper is structured as follows: Section II explains and motivates the technologies used in our framework, Section III presents the multicast framework, Section IV gives the evaluation of an implementation of this framework, Section V discusses related work and Section VI concludes.

II. Problem and motivation

A. IPv6 and Multicast

IPv6 has become the lingua franca of the IoT due to its standardisation, large address space and flexible address configuration schemes. However, IPv6 poses several challenges for the IoT due to the resource-constrained nature of IoT nodes. The de-facto radio technology for low-power networks is the IEEE 802.15.4 [7] standard that limits the maximum transmission unit to only 127 octets. As the minimum IPv6 packet size is 1280 octets, this necessitates an adaptation layer in order to be used over 802.15.4-based networks. To fill this gap, industry has defined 6LoWPAN [8] which fragments

IPv6 packets into multiple 802.15.4 frames and defines header compression schemes in order to minimize the overhead. We refer to 802.15.4-based networks with IPv6 enabled through the 6LoWPAN adaptation layer as LoWPANs.

Multicast support. IPv6 provides native support for multicast by defining a multicast-specific address structure. A packet sent to a multicast address is duplicated by network routers and forwarded to all members of the multicast group. Multicast alleviates the sender from transmitting a packet per recipient in the group, allowing for scalable *one-to-many* and *many-to-many* exchanges. It promotes decoupling between sender and receiver, since the sender only needs to know the group address instead of individual addresses. The price paid for these benefits is multicast group management and the overhead of multicast routing, which can cause additional communication and computation in the network.

Structure of an IPv6 multicast address. Multicast addresses are identified by a prefix, the IPv6 notation `ff00::8`. The next four bits, `flags`, determine if the address is permanently assigned or not. The following four flag bits, `scope`, determine the scope of the address (e.g., Link-Local). The remaining 112 bits are the group ID. Fig. 1 depicts the basic structure of an IPv6 multicast address.

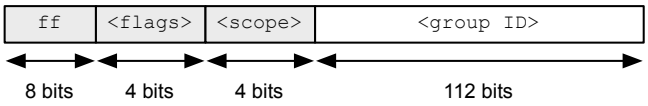

Fig. 1. Structure of an IPv6 multicast address.

Routing. In large LoWPANs, sender and receiver may be multiple radio hops away. The intermediate nodes on the path from sender to receiver are IP routers that examine the destination address and forward the packet as determined by its header. A routing protocol is in charge of filling the forwarding table at each node. The IETF routing standard for LoWPANs is RPL [5], which specifies how the network should build a Destination Oriented Directed Acyclic Graph (DODAG).

The DODAG structure of RPL natively supports multipoint-to-point traffic, as is common in sensor networks, but can also be used for point-to-point traffic between a pair of nodes in the network. Oikonomou et al. [6] proposed SMRF to extend RPL with support for efficient multicast. SMRF uses the existing DODAG structure of RPL and cross-layer optimizations to provide efficient multicast delivery. We leverage upon this work and use SMRF to meet the multicast routing requirements of our framework in LoWPANs.

B. Publish-subscribe middleware

Publish-subscribe middleware implements an interaction scheme with full decoupling of communicating entities in time, space and synchronisation [9]. Subscribers express their interest in specific information to *broker* services. Publishers of this information then simply push their content to the broker which asynchronously propagate it to all interested subscribers. This provides full decoupling: at no point in time do publishers or subscribers require any direct interaction with other pub-sub nodes. This model is an excellent fit for the scale and dynamics of the IoT. A variety of pub-sub architectures exist that range from completely centralised, through hierarchical to fully decentralised. Interactions in pub-sub systems happen at the application layer, where they appear as *one-to-one*, *one-to-many* and *many-to-many* communication.

At the network layer, these pub-sub interactions are mapped to *unicast* packet exchanges between hosts and brokers even when the published material is identical. Hence, the amount of packets necessary for a publication is directly proportional to the number of subscribers that are interested in the data. This limits scalability, and critically increases power consumption for all nodes that are involved in the routing of these packets.

In the following section, we propose a systematic framework that maps application-layer subscriber groups to network-layer multicast groups in order to reduce network overhead.

III. Multicast framework

IPv6 multicast communication requires two elements: (i.) a multicast group address and (ii.) support for managing group membership. The resource constraints of the IoT demand that this happens with low overhead and little additional complexity. Multicast addresses can be produced and managed in several ways. We propose two models to do so: a lightweight implicit multicast model and a more flexible explicit multicast model.

A. Lightweight multicast data distribution

The implicit multicast model enables very lightweight data distribution with almost zero configuration. It encodes an identifier of the published information and, optionally, a publisher address into the multicast group address itself. This identifier contains all metadata required for communication. For example, a publisher and subscriber might only be interested in communicating temperature readings within a group. In this case, the identifier encodes a temperature type into the multicast address. If a subscriber is interested in temperature readings of a specific publisher, the identifier encodes both the temperature type and the publisher address into the multicast address. In general, a publisher knows its own address and the information it wants to publish. Likewise, a subscriber knows the information it is interested in and, if relevant, from whom he wants to receive it.

The same multicast address can therefore be derived by both the publisher and the subscriber based on the encoding of a description of the information and the address of the publisher. This idea is captured in a customisable *derivation function* that takes as parameters a multicast prefix `prefix`, an information identifier `infId`, and node address `addr` in order to produce a multicast address:

$$\begin{aligned} MulticastAddress &= function(prefix, infId, addr) \\ &= prefix + infId + addr \end{aligned}$$

The derivation function compresses the node address, normally a (network) unique IPv6 unicast address, to fit the

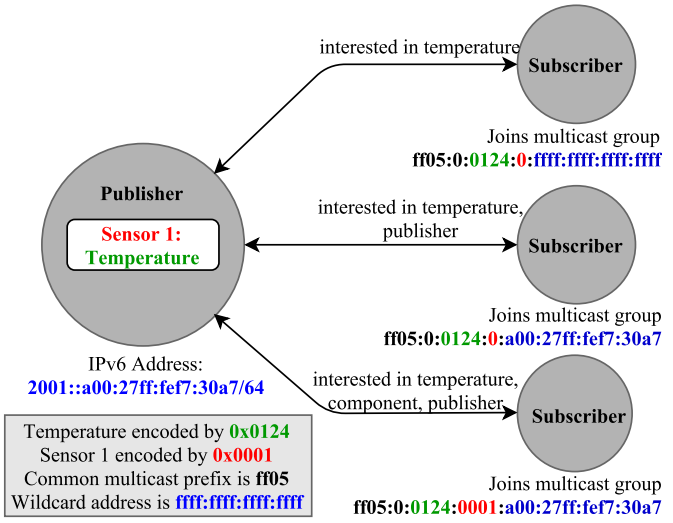

Fig. 2. Derivation of the same multicast address through encoding known information at publisher and subscriber nodes in the implicit model.

multicast address specification [10]. To maintain uniqueness of the address, this limits the scope of the multicast group to the network. Accordingly, of the 112-bits available (discounting 16-bits for the prefix), we need to reserve 64-bits in the multicast address for the node address. Thus there are 48-bits available to encode the information identifier, or around 10^{14} unique ids. With a suitable Machine-to-Machine (M2M) serialization scheme, such as the semantic LooCI type system [3] or the OMA lightweight M2M specification [11], we believe that this is sufficient for practical IoT applications.

In addition, there are 256 possible multicast prefixes. These can be used to determine Quality of Service (QoS) levels, dividing, for example, two multicast groups into one with reliable communications and one with unreliable communication. Fig. 2 illustrates various derivations for the LooCI case study, with a sensor component and event data type. Three cases are shown where a subscriber derives a multicast address based on its exact preferences. Because both nodes can autonomously derive the same address, we call this model *implicit*.

The implicit multicast model is lightweight and requires almost zero configuration. However, there are potential limits to the scalability, since all relevant metadata must be incorporated in the multicast addresses. Furthermore, this approach is insufficient for use cases in which there is no encoding scheme for the type of information a group wishes to communicate. For instance, if new hardware is used to acquire data, the user might become interested in a multicast group for that specific type of hardware. If the used derivation function cannot encode this information natively, implicit derivation becomes impossible. The implicit model is therefore best suited for smaller, constrained and dedicated use cases such as long-term, embedded home automation and monitoring systems. To deal with the limitations of this model for certain use cases, we have also developed an *explicit* multicast model that is more powerful and flexible.

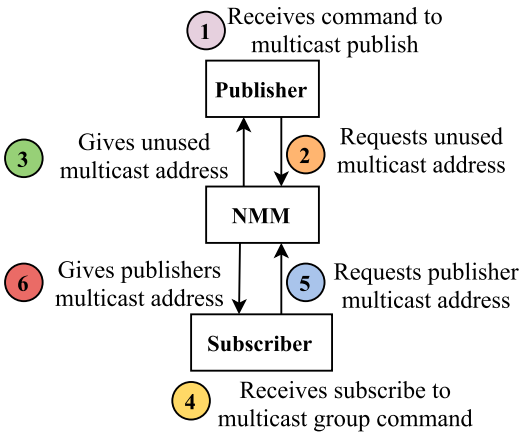

Fig. 3. Event flow of the establishment of a multicast group in the explicit model.

B. *Flexible multicast data distribution*

The explicit multicast model allows for extremely flexible and scalable data distribution by creating multicast groups explicitly instead of by derivation. This approach requires infrastructural support, e.g., an interface for the network administrator, to explicitly make the multicast groups. A multicast address can either be given by the administrator to each node, i.e., the *manual* case, or retrieved from a central entity, i.e., the *automatic* case. In the latter case, additional logic is required to map multicast addresses to application-level data types.

In the manual case, the node is provided with an address to use by the infrastructure and can thus join a multicast group. This introduces some additional communications to form the multicast group and also requires extra knowledge from the infrastructure but is powerful enough to allow all types of groups to be formed.

In the automatic case, shown in Fig. 3, a central entity, called the *Network Multicast Manager* (NMM) provides addresses. A publisher contacts the NMM with all relevant metadata and receives a multicast address in return. The NMM locally records the publisher's data and the assigned multicast address. In addition, a publisher can also request a used multicast address, associated to certain data (such as a publisher or type of information) to also multicast to that same group. A subscriber can then query the NMM for the desired group, e.g., a multicast group with a certain publisher or for temperature events, and is given the associated address. This enables flexible group communication, while lowering overhead on the network administrator. Due to its memory and network requirements, the NMM is expected to operate on a less resourced constrained device, such as a network gateway. This approach is better suited to large-scale, complex applications such as smart cities.

IV. Evaluation

We evaluated the implicit multicast model on real hardware to determine its influence on bandwidth consumption, energy

savings and memory requirements. The explicit model was evaluated but is not shown here since it simply adds an additional fixed overhead cost to establish membership to a multicast group. The evaluation gave similar results to the implicit model, corresponding with a one-time cost of 2 additional packets per node for each joined multicast group due to communications with the NMM.

The models are implemented on the LooCI pub-sub middleware [3], which is publicly available on https://distrinet.cs.kuleuven.be/software/looci. In the case of network gateways, we use the Java/OSGi version of LooCI on a regular computer, while for the IoT devices we use the Contiki variant [12] on the Zigduino platform. The Zigduino [13] is an Arduino-compatible mote based on the ATmega128RFA1 [14], which offers a 16 MHz MCU, 16 KB of RAM, 128 KB of Flash and an IEEE 802.15.4 radio. The OSGi and Contiki nodes operate on a one-hop Ethernet and LoWPAN network, respectively. The gateway OSGi publisher configures all the publishers and subscribers in the deployment. All tests were run for 15 minutes with publishers publishing an event every 10 seconds. All packets and their payload were logged locally at the nodes and on the gateway through a sniffer with Wireshark.

A. *Bandwidth*

Three different scales are considered with testbeds of: 7, 15 and 25 Zigduino nodes connected to a more powerful OSGi gateway. Each publisher has 5, 10 and 20 subscribers each.

We compare the bandwidth consumption of LooCI with multicast against that of LooCI with unicast. The traffic in the network is measured as total bandwidth consumption in kilobytes and can be categorised into UDP payload traffic and ICMPv6 overhead traffic.

Fig. 4 shows the bandwidth consumption for a gateway OSGi publisher for the different scales. The bandwidth consumed by multicast is invariant to the amount of nodes to which data is being published: all scales have a constant slope, in contrast to unicast where the slope varies in proportion to the number of subscribers. In addition, the initial network configuration effort is strongly reduced since publishers only need to be configured for publishing to a multicast address instead of to a list of subscribers.

Fig. 5 shows the bandwidth consumption for a Contiki publisher. Multicast is no longer invariant to the amount of nodes, due to multicast overhead in the LoWPAN, but the slope increases less rapidly than for unicast. Specifically, doubling the amount of subscribers doubles the gradient of the slope for unicast while it only increases by roughly half for multicast. Overall, total bandwidth consumption over 15 minutes by a Contiki publisher decreases by up to 54% for 10 subscribers and 66% for 20 subscribers when using multicast instead of unicast in this scenario. These results make a strong case for supporting application layer pub-sub with multicast as precious bandwidth and, critically, energy are conserved by eliminating redundant transmission.

Fig. 6 shows the total network traffic (in consumed bandwidth) detected by a sniffer on the Ethernet and LoWPAN

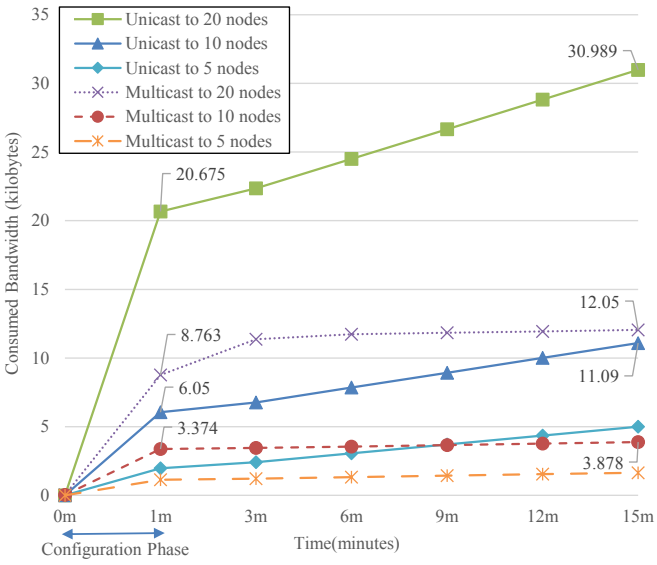

Fig. 4. Payload bandwidth consumption in time by an OSGi publisher for three different scales. Multicast shows reduced configuration effort and invariant slope in contrast to unicast.

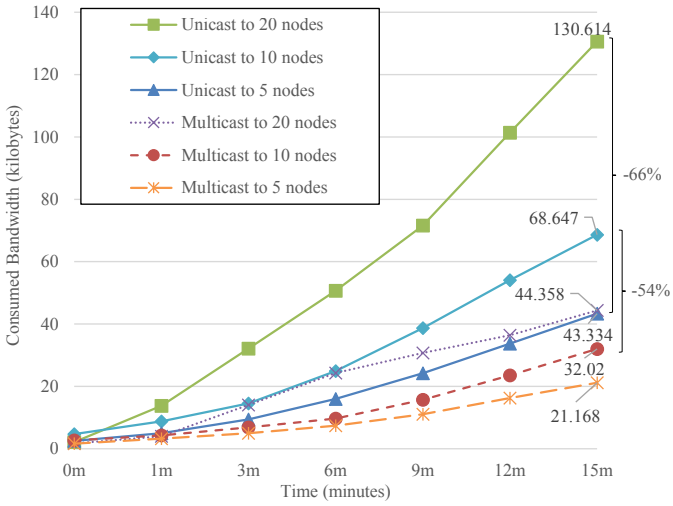

Fig. 5. Total, payload and overhead, bandwidth consumption in time by a Contiki publisher for three different scales. Multicast shows better scaling with a reduction in bandwidth consumption over 15 minutes of 54% for 10 subscribers and 66% for 20 subscribers.

networks, where a multicast approach consumes up to six times less bandwidth. Similar to the publisher analysis, these bandwidth savings increase with the size of the network which ensures more scalable communications.

Multicast routing causes some overhead in the RPL network. To better determine this cost, we measured the total traffic in the LoWPAN in a dynamic test testbed in which nodes were periodically added, up to 13 total. Fig. 7 shows the total ICMPv6 overhead and UDP payload over 8 minutes in the dynamic network. Multicast has a higher overhead (approximately 50% more), but a much lower payload cost, around one-fifth thereby making communications cheaper

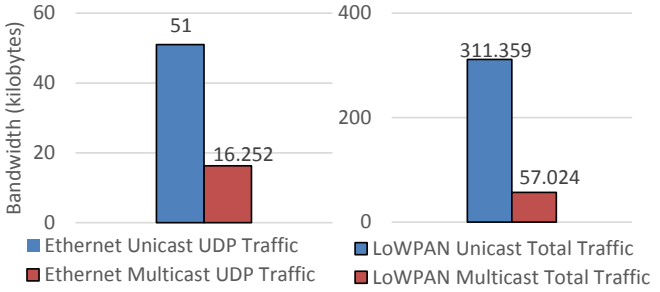

Fig. 6. Consumed bandwidth on a 25 node Ethernet and LoWPAN network for unicast and multicast. Multicast consumes up to six times less bandwidth.

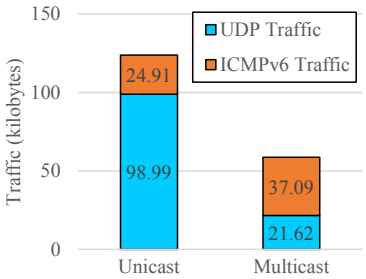

Fig. 7. Breakdown of the traffic in a dynamic LoWPAN into payload (UDP) and overhead (ICMPv6) for the unicast and multicast scenario. Multicast compensates increased overhead with a reduction in total payload.

overall. These results show that multicast quickly compensates its added overhead, even in dynamic scenarios.

B. *Energy savings*

Multicast saves energy by reducing the amount of packets sent by a publisher. In unicast, a publisher has to individually send a packet to each subscriber while for multicast it has to do this only once. A Contiki Zigduino node using Contiki X-MAC requires, on average, 21.41 mJ per standard frame transmission. We estimate the energy cost of the transmissions by multiplying the above average cost with the amount of packets sent. Fig. 8 shows the estimated energy consumption in a fixed time period for a Contiki publisher for three different scales with unicast and multicast. Unicast needs more energy as the scale increases, up to 37 J after 15 minutes, while multicast remains invariant to the scale, needing about 2 J for all scales in the same time period.

C. *Memory footprint*

The implementation of multicast functionality incurs an additional memory cost. We analyse this for the resource-constrained Zigduino platform using LooCI Contiki (LC). Table I shows the absolute memory used (bytes) and the proportion of available capacity that is used (%).

Our implementation requires 6172 bytes of flash memory and at least 204 bytes RAM. This corresponds to an increased usage of, respectively, 4.7% and 1.3% of the total capacity. At runtime, multicast group membership requires negligibly more RAM than regular pub-sub connections, about one byte per joined group. However, multicast enables using less

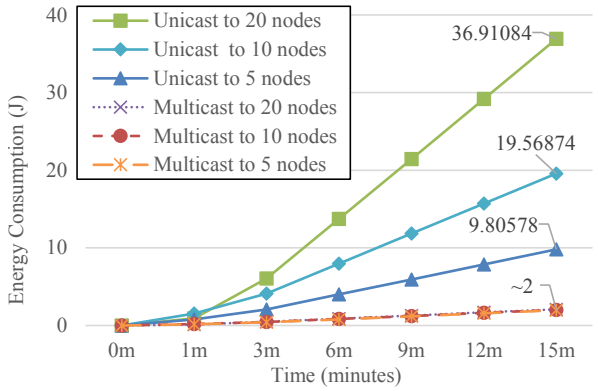

Fig. 8. Estimated energy consumption of transmissions for a Contiki publisher on the Zigduino platform in a fixed time period for unicast and multicast. Multicast is invariant, unicast scales proportionally to the amount of subscribers.

connections, so that total RAM usage decreases. These results confirm that multicast can be implemented with a reasonable cost in increased memory footprint.

TABLE I
PROGRAM CODE SIZE IN FLASH MEMORY AND RAM IN BYTES FOR UNICAST AND MULTICAST LOOCI AND THEIR DIFFERENCE.

Memory(bytes)	Unicast LC	Multicast LC	Difference
Flash Memory	68822 (52.5%)	74994 (57.2%)	6172 (Δ4.7%)
RAM	11661 (71.2%)	11865 (72.5%)	204 (Δ1.3%)

V. RELATED WORK

Król et al. proposed *Featurecast* [15] in which addressing and routing are based on node features defined as predicates. Each node constructs its address from the set of its features and disseminates it in the network so that intermediate nodes can build routing tables. In this way, a node can send a packet to a set of nodes matching given features. Their analysis compares this system with a home-made implementation of IP multicast for the Contiki platform. They show that Featurecast has a lower memory footprint than multicast at large scales. Accordingly, Featurecast is better suited to applications that require very large numbers of groups. We view this work as complementary and believe that it could fulfil the same role in our architecture as multicast.

LightPS, proposed by Ahulló, López and Skarmeta, [16] is a framework for lightweight content-based publish-subscribe systems. It avoids explicit multicast group management or multicast tree maintenance by mapping subscription and event information to keys. Their routing algorithm and an underlying DHT infrastructure enable every mapped key to reach its owner node. This solution also maps subscriptions and encodes information to enable lightweight management and routing, but they remain strictly within the application layer and focus more on big p2p networks.

Banavar et al. [17] discusses the importance of efficient multicast communications in pub-sub systems and how to

match multicast events to groups of subscribers. To do so, they propose *link matching*, an algorithm to match and forward events to content-based subscribers. This work validates the approach of our paper but there are a number of significant differences. Specifically, their research only considers IPv4 multicast for powerful devices in a static configurations, which is very different from typical IoT scenarios.

The work of Sano et al. [18] focuses on an application-level multicast approach. Their scheme improves the performance penalties of application-level multicast by making use of network support. They improve the transmission performance by exploiting network-level information, such as delivery path information. It is similar to our work, but we specifically take the IoT into account and we map the pub-sub paradigm (an application-level multicast) to the network level instead of exploiting network-level knowledge in the application.

Chen et al. [19] developed a publish-subscribe QoS aware middleware for WSNs, *PS-QUASAR*. In this model, all nodes in the network are potential publishers of each topic. The model leverages multicast techniques to lower energy consumption and improve delivery ratios. They developed an entire middleware solution instead of complementing an existing one, as we did. Therefore, their work is not applicable to existing systems, such as LooCI. In contrast, our approach is generic and can be applied across any pub-sub middleware.

VI. Conclusions and future work

The IoT consists of large-scale networks of resource-constrained devices. Therefore, efficient and flexible communications are necessary. This paper contributes a generic approach to mapping pub-sub middleware onto IPv6 multicast in order to reduce network overhead. The resulting multicast framework embeds two concrete models: a less flexible but lightweight implicit multicast model and a versatile but costlier explicit multicast model.

Evaluation shows that our framework improves scalability, reduces overall network traffic and achieves better energy efficiency. Bandwidth consumption of a multicasting sensor decreases by up to 54% for 10 subscribers and 66% for 20 subscribers. These benefits are obtained with a limited memory footprint, corresponding to an additional usage of 4.7% of the total flash memory and 1.3% of the total RAM. This proves the approach is both feasible and beneficial for the IoT.

As future work, we believe that the derivation function and encoding scheme for the implicit model are interesting to investigate further. Currently, a fixed prefix, the information identifier and publisher address are mapped into the multicast address, which leaves plenty of bits left. The function itself or any unused bits could be used to support different QoS levels, to encode other information or to obtain other benefits. For example, we are investigating hash functions to encode the information compactly and make more flexible groups.

Acknowledgment

This research is partially funded by the Research Fund KU Leuven and iMinds, and is conducted in the context of the EMD project.

References

[1] J. Ceclio and P. Furtado, "Existing Middleware Solutions for Wireless Sensor Networks," in *Wireless Sensors in Heterogeneous Networked Systems*, ser. Computer Communications and Networks. Springer International Publishing, 2014, pp. 39–59.

[2] B. Bhuyan, H. K. D. Sarma, and N. Sarma, "A survey on middleware for wireless sensor networks," *Journal of Wireless Networking and Communications*, vol. 4, no. 1, pp. 7–17, 2014.

[3] D. Hughes, K. Thoelen, J. Maerien, N. Matthys, J. Del Cid, W. Horre, C. Huygens, S. Michiels, and W. Joosen, "LooCI: The Loosely-coupled Component Infrastructure," in *Network Computing and Applications (NCA), 2012 11th IEEE International Symposium on*, Aug 2012, pp. 236–243.

[4] T. Sheltami, A. Al-Roubaiey, A. Mahmoud, and E. Shakshuki, "A publish/subscribe middleware cost in wireless sensor networks: A review and case study," in *Electrical and Computer Engineering (CCECE), 2015 IEEE 28th Canadian Conference on*, May 2015, pp. 1356–1363.

[5] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," Internet Requests for Comments, RFC 6550, March 2012.

[6] G. Oikonomou, I. Phillips, and T. Tryfonas, "IPv6 Multicast Forwarding in RPL-Based Wireless Sensor Networks," *Wirel. Pers. Commun.*, vol. 73, no. 3, pp. 1089–1116, Dec. 2013.

[7] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks," Internet Requests for Comments, RFC 4944, September 2007.

[8] N. Kushalnagar, G. Montenegro, and C. Schumacher, "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals," Internet Requests for Comments, RFC 4919, August 2007.

[9] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The Many Faces of Publish/Subscribe," *ACM Comput. Surv.*, vol. 35, no. 2, pp. 114–131, Jun. 2003.

[10] R. Hinden and S. Deering, "IP Version 6 Addressing Architecture," Internet Requests for Comments, RFC 4291, February 2006.

[11] L. Tian, "Lightweight M2M (OMA LWM2M)," *OMA device management working group (OMA DM WG), Open Mobile Alliance (OMA)*, 2012.

[12] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki - a lightweight and flexible operating system for tiny networked sensors," in *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*, Nov 2004, pp. 455–462.

[13] L. Electromechanical. Zigduino R2 Manual. Available: www.wiki.logos-electro.com/zigduino-r2-manual [Accessed: October 2015].

[14] Atmel. AVR ATmega128RFA1 Datasheet. Available: http://www.atmel.com/Images/Atmel-8266-MCU_Wireless-ATmega128RFA1_Datasheet.pdf [Accessed: October 2015].

[15] M. Król, F. Rousseau, and A. Duda, "Featurecast: Lightweight Data-Centric Communications for Wireless Sensor Networks," in *Wireless Sensor Networks*. Springer, 2015, pp. 202–217.

[16] J. Ahullo, P. Lopez, and A. Gomez Skarmeta, "LightPS: Lightweight Content-Based Publish/Subscribe for Peer-to-Peer Systems," in *Complex, Intelligent and Software Intensive Systems, 2008. CISIS 2008. International Conference on*, March 2008, pp. 342–347.

[17] G. Banavar, T. Chandra, B. Mukherjee, J. Nagarajarao, R. Strom, and D. Sturman, "An efficient multicast protocol for content-based publish-subscribe systems," in *Distributed Computing Systems, 1999. Proceedings. 19th IEEE International Conference on*, 1999, pp. 262–272.

[18] T. Sano, T. Noguchi, and M. Yamamoto, "Improving efficiency of application-level multicast with network support," in *Group Communications and Charges. Technology and Business Models*. Springer, 2003, pp. 13–22.

[19] J. Chen, M. Díaz, B. Rubio, and J. M. Troya, "PS-QUASAR: A publish/subscribe QoS aware middleware for Wireless Sensor and Actor Networks," *Journal of Systems and Software*, vol. 86, no. 6, pp. 1650–1662, 2013.